US005838906A

United States Patent [19]

Doyle et al.

[11] **Patent Number:** 5,838,906

[45] **Date of Patent:** Nov. 17, 1998

[54] **DISTRIBUTED HYPERMEDIA METHOD FOR AUTOMATICALLY INVOKING EXTERNAL APPLICATION PROVIDING INTERACTION AND DISPLAY OF EMBEDDED OBJECTS WITHIN A HYPERMEDIA DOCUMENT**

[75] Inventors: **Michael D. Doyle**, Alameda; **David C. Martin**, San Jose; **Cheong S. Ang**, Pacifica, all of Calif.

[73] Assignee: **The Regents of the University of California**, Oakland, Calif.

[21] Appl. No.: **324,443**

[22] Filed: **Oct. 17, 1994**

[51] **Int. Cl.**[6] **C06F 9/44**; C06F 15/16; C06F 17/30

[52] **U.S. Cl.** **395/200.32**; 395/200.28; 395/680; 395/685; 345/326; 345/346; 707/501; 707/513; 707/515; 707/516

[58] **Field of Search** 395/157, 200.03, 395/161, 118, 144, 145, 146, 147, 148, 683, 777, 778, 762, 326, 333, 334, 335, 676, 682, 685, 684, 200.32, 200.33, 200.47–200.49; 707/501, 513, 515, 516; 345/326, 343, 346

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,815,029	3/1989	Barker et al.	707/516
4,847,604	7/1989	Doyle	340/706
4,949,248	8/1990	Caro	395/200.03
5,146,553	9/1992	Noguchi et al.	707/516
5,202,828	4/1993	Vertelney et al.	364/419
5,204,947	4/1993	Bernstein et al.	395/157
5,206,951	4/1993	Khoyi et al.	395/683
5,274,821	12/1993	Rouquie	395/705
5,307,499	4/1994	Yin	395/700
5,321,806	6/1994	Meinerth et al.	395/162
5,321,808	6/1994	Rupp et al.	395/164
5,347,632	9/1994	Filepp et al.	395/200.09

(List continued on next page.)

OTHER PUBLICATIONS

Stephen Le Hunte, "<EEMBED>—Embedded Objects", HTML Reference Library—HTMLIB v2.1, 1995: n.pag. Online. Internet.

"A Little History of the world Wide Web", n.pag. Online. Internet: available http://www.w3.org/History.html.

"NCSA Mosaic Version Information", n.pag. Online. Internet: available http://www.ncsa.uiuc.edu/SDG/Software.

"The second phase of the revolution", WIRED, Oct. 1994, pp. 116–152.

(List continued on next page.)

Primary Examiner—Dinh C. Dung

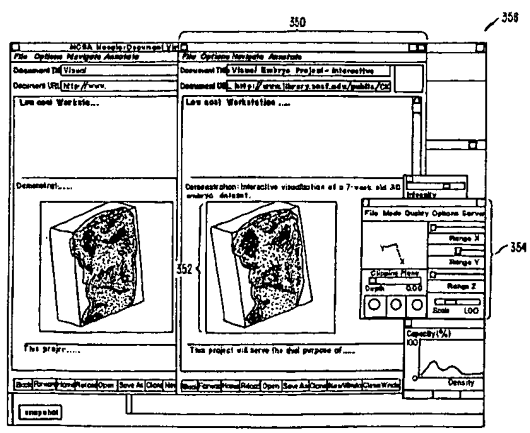
Attorney, Agent, or Firm—Townsend and Townsend and Crew LLP

[57] **ABSTRACT**

A system allowing a user of a browser program on a computer connected to an open distributed hypermedia system to access and execute an embedded program object. The program object is embedded into a hypermedia document much like data objects. The user may select the program object from the screen. Once selected the program object executes on the user's (client) computer or may execute on a remote server or additional remote computers in a distributed processing arrangement. After launching the program object, the user is able to interact with the object as the invention provides for ongoing interprocess communication between the application object (program) and the browser program. One application of the embedded program object allows a user to view large and complex multi-dimensional objects from within the browser's window. The user can manipulate a control panel to change the viewpoint used to view the image. The invention allows a program to execute on a remote server or other computers to calculate the viewing transformations and send frame data to the client computer thus providing the user of the client computer with interactive features and allowing the user to have access to greater computing power than may be available at the user's client computer.

10 Claims, 9 Drawing Sheets

Microfiche Appendix Included
(4 Microfiche, 375 Pages)



U.S. PATENT DOCUMENTS

5,367,635	11/1994	Bauer et al.	395/200.32
5,390,314	2/1995	Swanson	395/500
5,418,908	5/1995	Keller et al.	395/200.32
5,544,320	8/1996	Konrad	395/200.09
5,581,686	12/1996	Koppolu et al.	395/340
5,606,493	2/1997	Duscher et al.	395/200.32
5,652,876	7/1997	Ashe et al.	707/516

OTHER PUBLICATIONS

Vetter, Ronald "Mosaic and the World–Wide Web," Computer Magazine, v.27, Iss.10, pp. 49–57, Oct. 1994.

Wynne et al. "Lean Management, Group Support Systems, and Hypermedia: a Combination Whose Time Has Come," System Sciences, 1993 Anuall Hawaii Int'l Conf., pp. 112–121.

Hansen, Wilfred "Andrew as a Multiparadigm Environment for Visual Languages," Visual Languages, 1993 IEEE Symposium, pp. 256–260.

Moran, Patrick "Tele–Nicer–slicer–Dicer: A New Tool for the Visualization of Large Volumetric Data", NCSA Technical Report (TR014), Aug. 1993.

Berners–Lee "Hypertext Markup Language (HTML)", HTML Internet Draft, IIIR working Group, Jun. 1993.

University of Southern California's Mercury Project—"USC Mercury Project:Interface", Project Milestones, USC Press Release—obtained from Internet, http://www.usc.edu/dept/raiders/.

Hansen, Wilfred "Enhancing documents with embedded programs: How Ness extends in the Andrew ToolKit", IEEE Computer Language, 1990 International Conference.

Tani, M., et al., "Object–Oriented Video: Interaction with Real–World Objects Through Live Video", May 1992, p. 593–598.

Crowley, T., et al., "MMConf: An Infrastructure for Building Shared Multimedia Applications", CSCW 90 Proceedings, Oct. 1990, p. 329–342.

Davis, H., et al., "Towards An Integrated Information Environment With Open Hypermedia System", ACM ECHT Conference, Dec. 1992, pp. 181–190.

Ferrara, F., "The KIM Query System", Abstract, SIGCHI Bulletin, vol. 6, No. 3, Jul. 1994, pp. 30–39.

Gibbs, S., "Composite Multimedia and Active Objects", OOPSLA '91, pp. 97–112.

Davis, H., et al., "Microcosm: An Open Hypermedia System", Interchi '93, Apr. 1993, p. 526.

Vaziri, A., "Scientific Visualization in High–Speed Network Environments", Computer Networks and ISDN Systems 22, 1991, pp. 111–129.

Cullen, J., et al., "The Use of FTAM to access graphical pictures across wide area networks", Computer Networks and ISDN Systems, 1992, pp. 337–383.

Lashkari, Y.Z., et al., "PLX: A Proposal to Implement a General Broadcasting Facility in a Distributed Environment Running X Windows", Comput. & Graphics, vol. 16, No. 2, pp. 143–149, 1992.

Kirste, T., "Spacepicture—An Interactive Hypermedia Satellite Image Archival System", Comput. & Graphics, vol. 17, No. 3, pp. 251–260, 1993.

Coulson, G., et al., "Extensions to ANSA for Multimedia Computing", Computers Networks and ISDN Systems 25, 1992, pp. 305–323.

Huynh, Duong Le, et al., "PIX: An Object–Oriented Network Graphics Environment", Comput. & Graphics, vol. 17, No. 3, pp. 295–304, 1993.

Berners–Lee, T.J., et al., The World–Wide Web, Computer Networks and ISDN Systems 25, 1992, pp. 454–459.

Shackelford, D.E., et al., "The Architecture and Implementation of a Distributed Hypermedia Storage System", Hypertext '93 Proceedings, Nov. 1993, pp. 1–13.

Labriola, D., "Remote Possibilities", PC Magazine, Jun. 14, 1994, pp. 223–228.

Udell, J., "Visual Basic Custom Controls Meet OLE", Byte Magazine, Mar. 1994, pp. 197–200.

Sarna, D.E., et al., "OLE Gains Without (Much) Pain", Datamation Magazine, Jun. 15, 1994, pp. 31 and 113.

Rizzo, J., "What's OpenDoc?", MacUser magazine, Apr. 1994, pp. 119–123.

Fogarty, K., et al., "Microsoft's OLE can be network Trojan Horse", Network World Magazine, Jun. 27, 1994, vol. 11, No. 26, pp. 1 and 75.

"Cello WWW Browser Release 1.01a", Article obtained from the Internet, ftp.law.cornell.edu/pub/L11/Cello no DDE, Mar. 16, 1994, pp. 2–9.

"OLE 2.0: Death to Monoliths", Byte Magazine, Mar. 1994, p. 122.

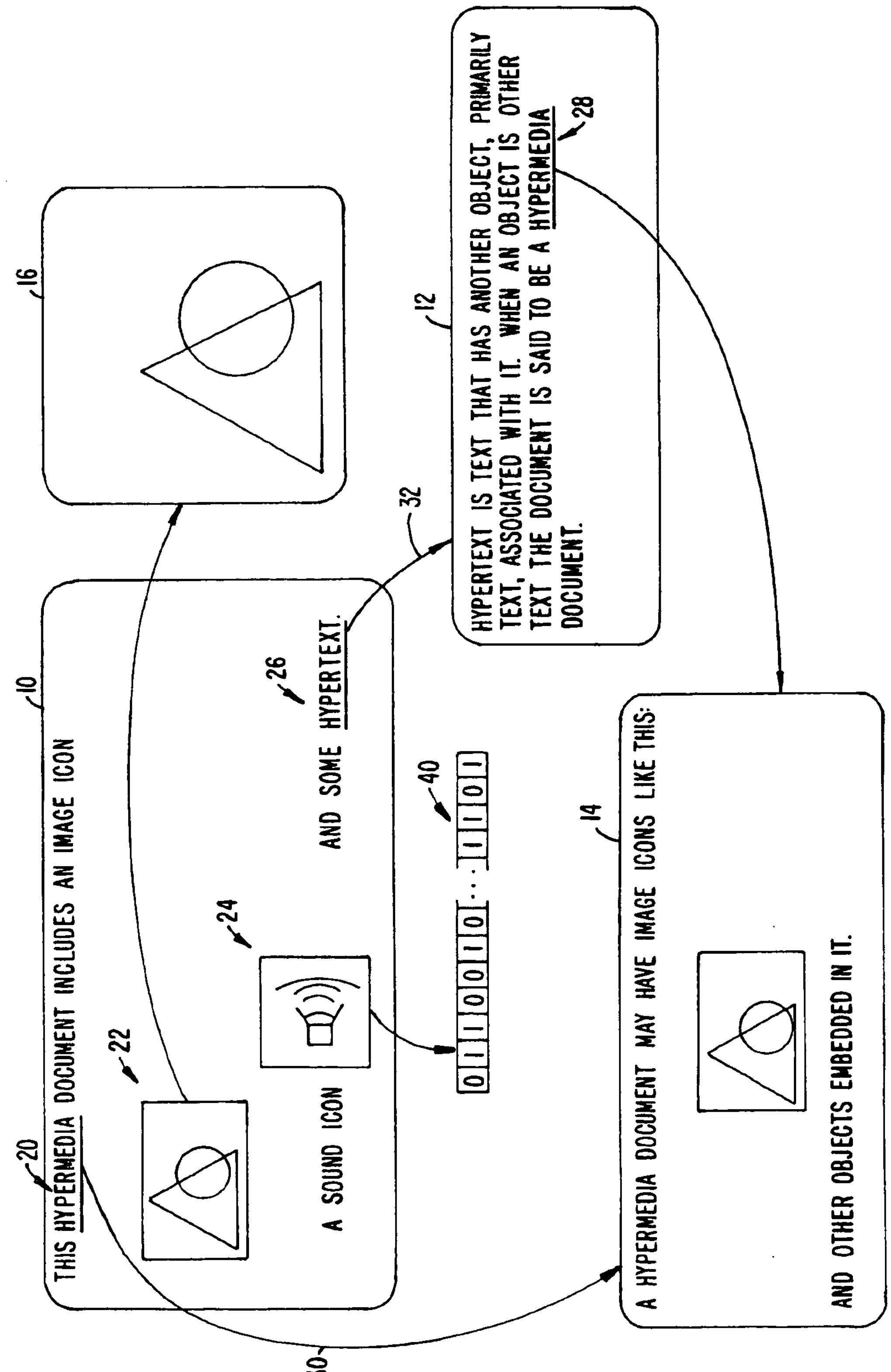


FIG. 1. PRIOR ART

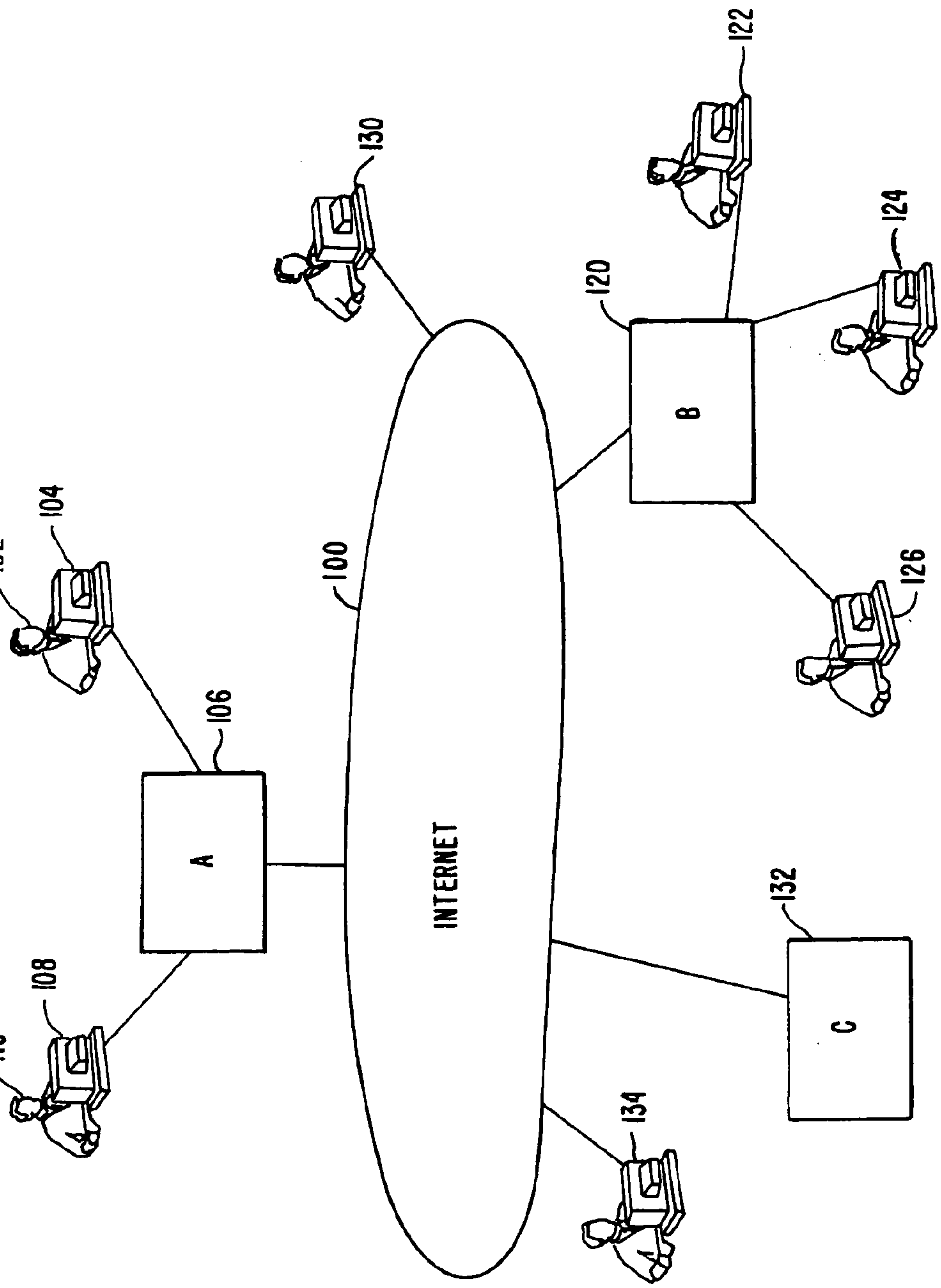


FIG. 2. PRIOR ART

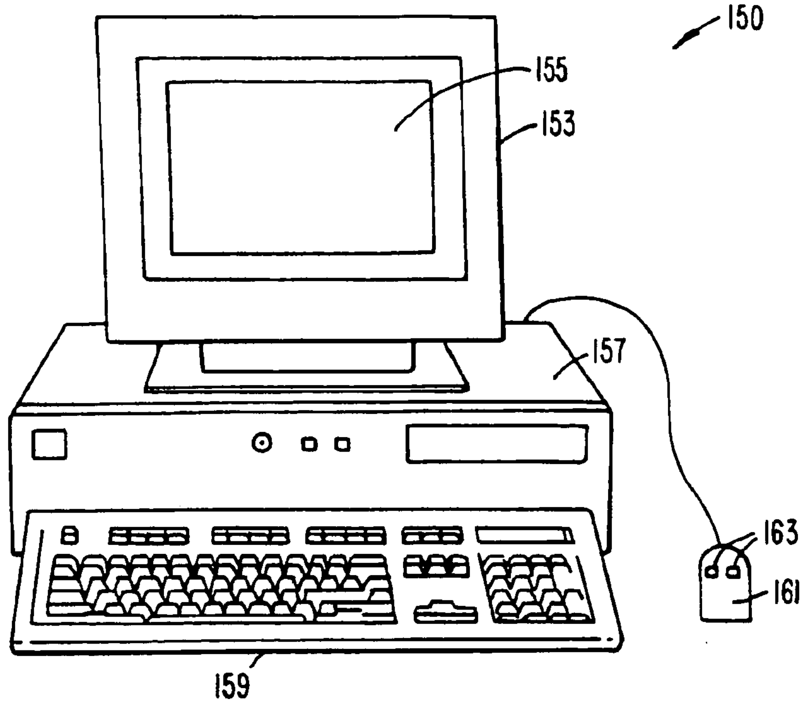

150
155
153
157
163
161
159

FIG. 3.

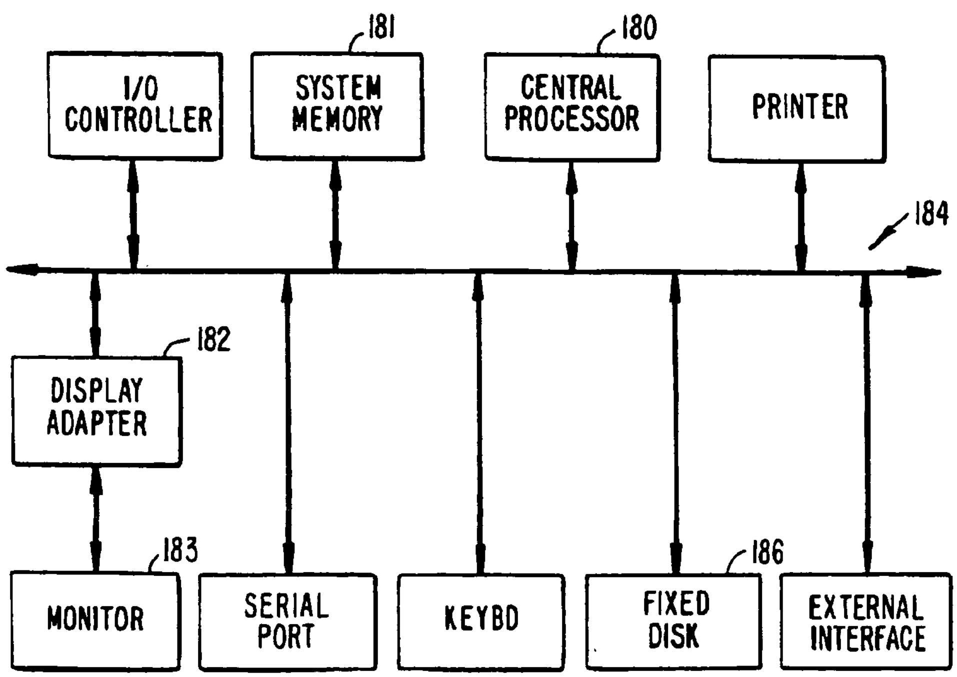

181
180
I/O CONTROLLER
SYSTEM MEMORY
CENTRAL PROCESSOR
PRINTER
184
182
DISPLAY ADAPTER
183
186
MONITOR
SERIAL PORT
KEYBD
FIXED DISK
EXTERNAL INTERFACE

FIG. 4.

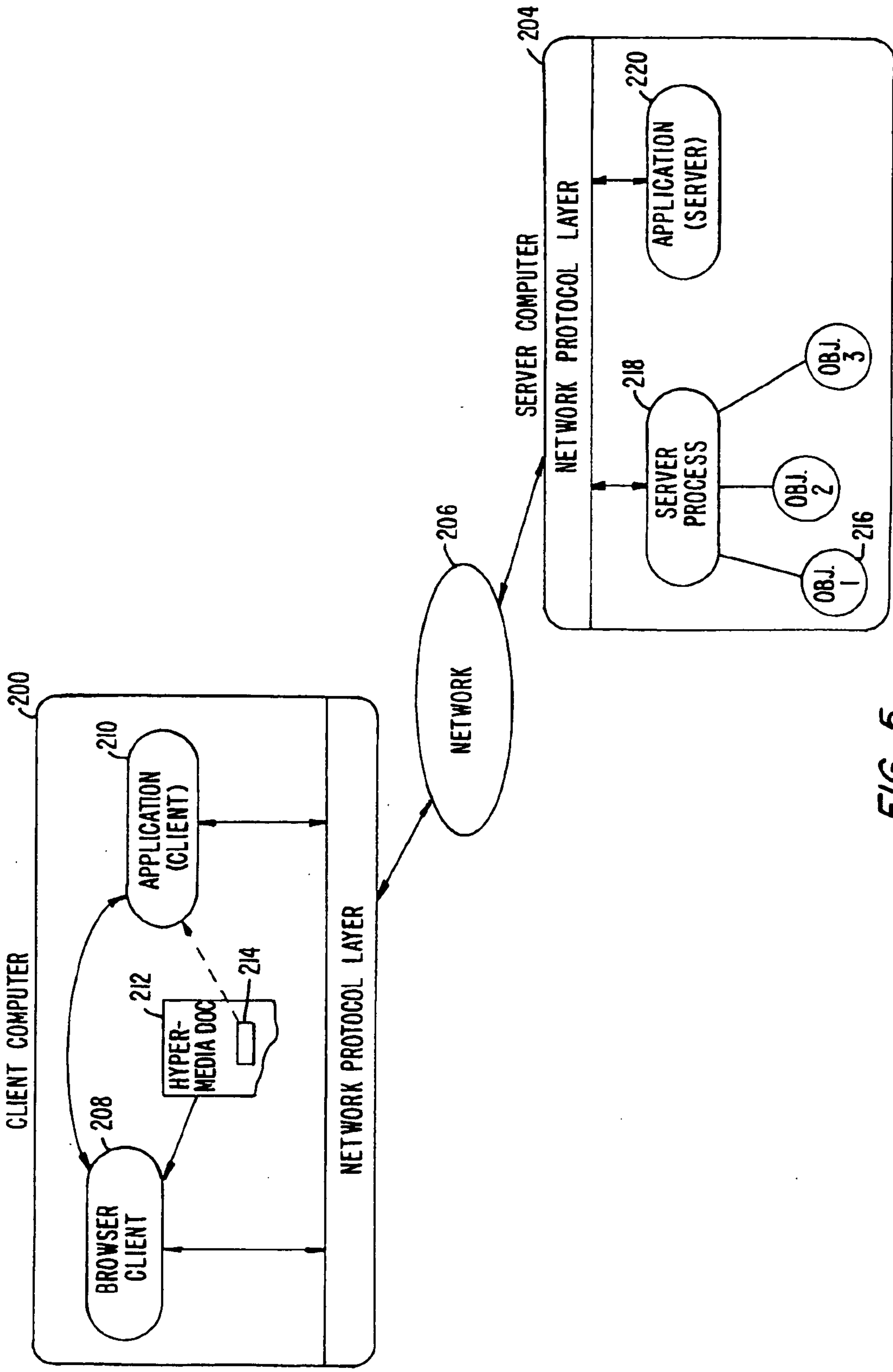
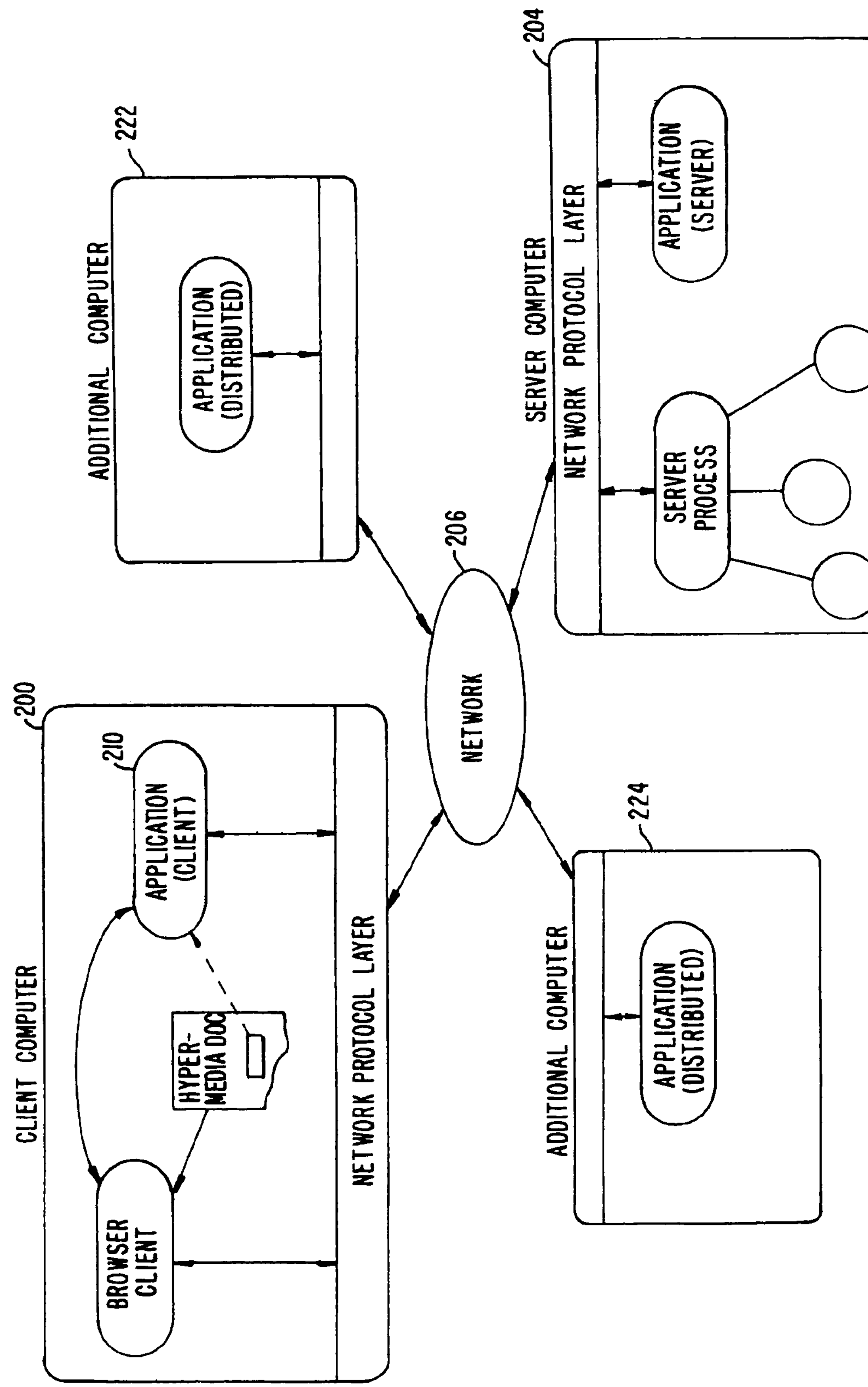

204
SERVER COMPUTER
NETWORK PROTOCOL LAYER
220
APPLICATION (SERVER)
218
SERVER PROCESS
OBJ. 1
OBJ. 2
OBJ. 3
216
206
NETWORK
200
CLIENT COMPUTER
210
APPLICATION (CLIENT)
212
HYPER-MEDIA DOC
214
208
BROWSER CLIENT
NETWORK PROTOCOL LAYER

FIG. 5



CLIENT COMPUTER
200
210
APPLICATION (CLIENT)
BROWSER CLIENT
HYPER-MEDIA DOC
NETWORK PROTOCOL LAYER
NETWORK
206
ADDITIONAL COMPUTER
222
APPLICATION (DISTRIBUTED)
SERVER COMPUTER
204
NETWORK PROTOCOL LAYER
APPLICATION (SERVER)
SERVER PROCESS
ADDITIONAL COMPUTER
224
APPLICATION (DISTRIBUTED)

FIG. 6

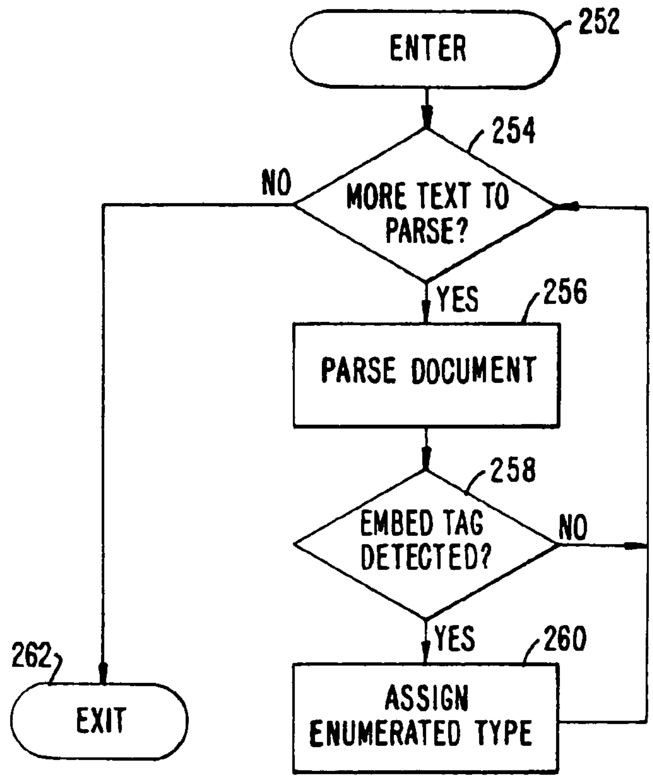

ENTER
252
MORE TEXT TO PARSE?
254
NO
YES
PARSE DOCUMENT
256
EMBED TAG DETECTED?
258
NO
YES
ASSIGN ENUMERATED TYPE
260
262
EXIT

FIG. 7A.

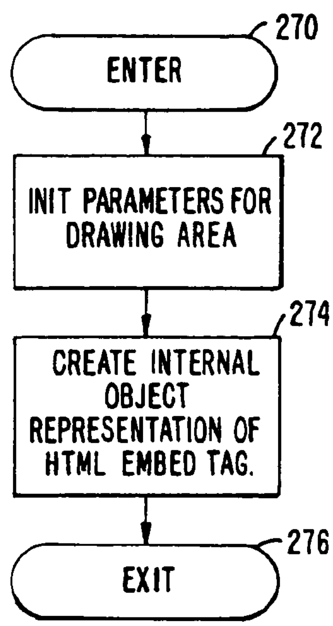

ENTER
270
INIT PARAMETERS FOR DRAWING AREA
272
CREATE INTERNAL OBJECT REPRESENTATION OF HTML EMBED TAG.
274
EXIT
276

FIG. 7B.

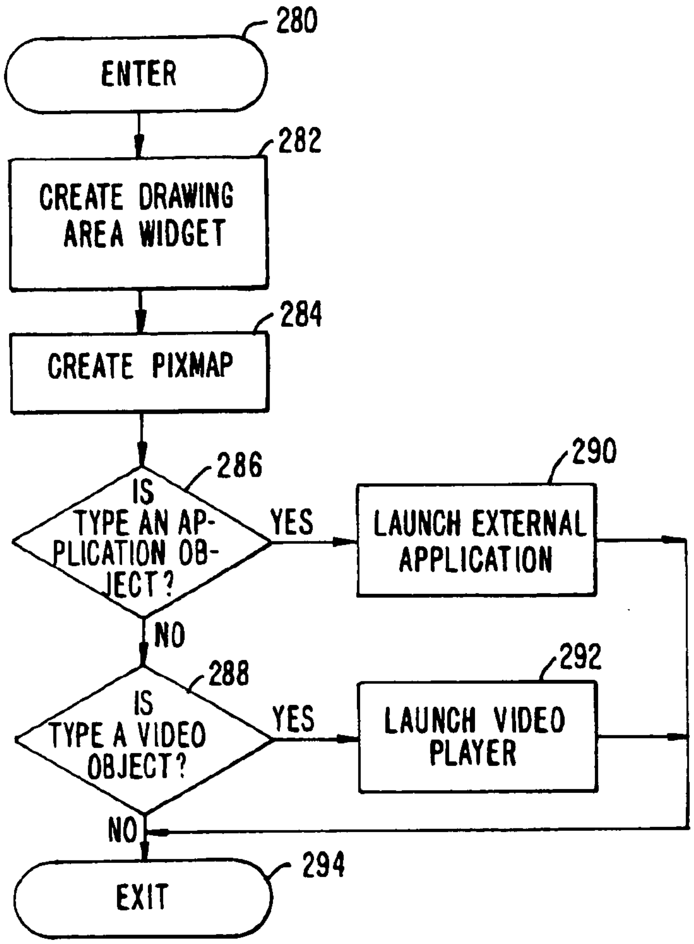

280
ENTER
282
CREATE DRAWING AREA WIDGET
284
CREATE PIXMAP
286
IS TYPE AN APPLICATION OBJECT?
YES
290
LAUNCH EXTERNAL APPLICATION
NO
288
IS TYPE A VIDEO OBJECT?
YES
292
LAUNCH VIDEO PLAYER
NO
294
EXIT

FIG. 8A.

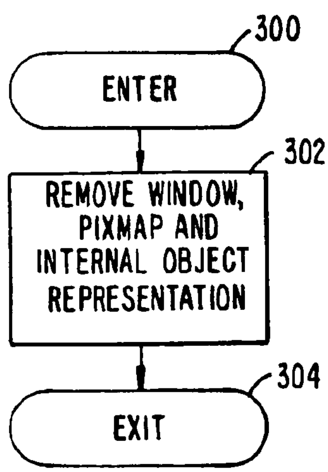
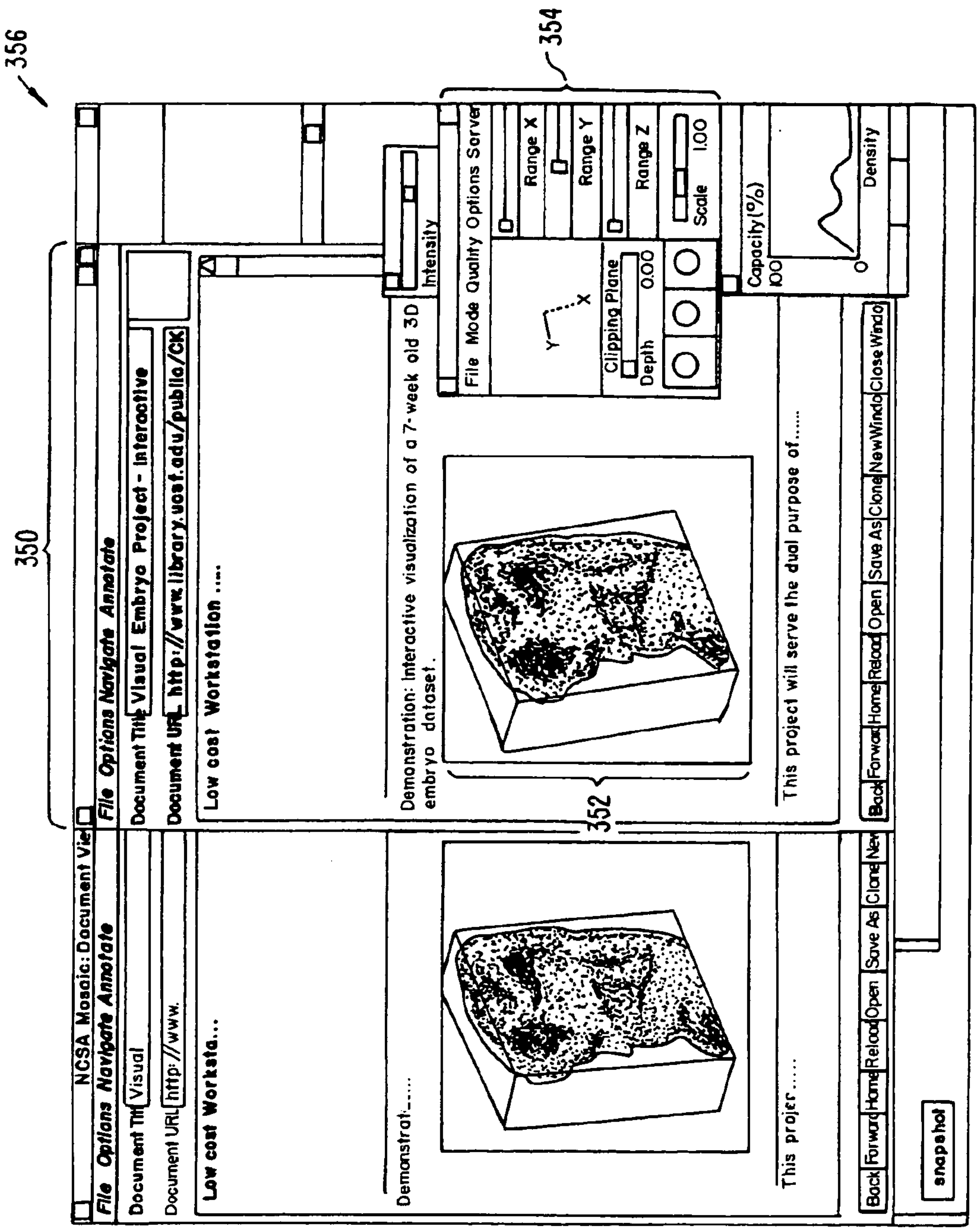

300
ENTER
302
REMOVE WINDOW, PIXMAP AND INTERNAL OBJECT REPRESENTATION
304
EXIT

FIG. 8B.



356
350
352
354
NCSA Mosaic: Document Vie
File Options Navigate Annotate
Document Title Visual Embryo Project - Interactive
Document URL http://www.library.ucsf.edu/publio/CK
Low cost Workstation ....
Demonstration: Interactive visualization of a 7-week old 3D embryo dataset.
This project will serve the dual purpose of.....
File Mode Quality Options Server
Range X
Range Y
Range Z
Clipping Plane
Depth
Scale
Intensity
Capacity(%)
Density
Back Forward Home Reload Open Save As Clone NewWindow CloseWindow
snapshot

FIG. 9.

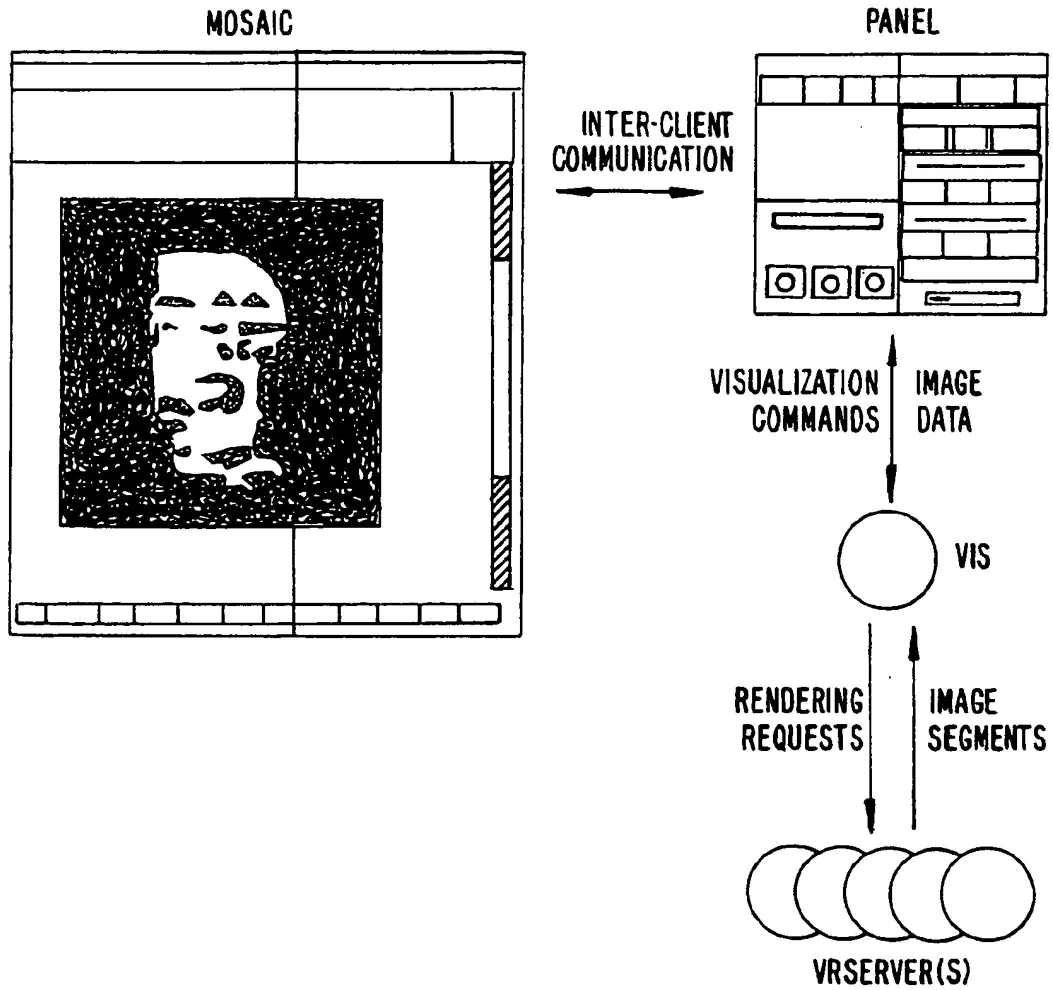

MOSAIC
PANEL
INTER-CLIENT
COMMUNICATION
VISUALIZATION
COMMANDS
IMAGE
DATA
VIS
RENDERING
REQUESTS
IMAGE
SEGMENTS
VRSERVER(S)

FIG. 10.

DISTRIBUTED HYPERMEDIA METHOD FOR AUTOMATICALLY INVOKING EXTERNAL APPLICATION PROVIDING INTERACTION AND DISPLAY OF EMBEDDED OBJECTS WITHIN A HYPERMEDIA DOCUMENT

NOTICE REGARDING COPYRIGHTED MATERIAL

BACKGROUND OF THE INVENTION

This invention relates generally to manipulating data in a computer network, and specifically to retrieving, presenting and manipulating embedded program objects in distributed hypermedia systems.

Computer networks are becoming increasingly popular as a medium for locating and accessing a wide range of data from locations all over the world. The most popular global network is the Internet with millions of computer systems connected to it. The Internet has become popular due to widely adopted standard protocols that allow a vast interconnection of computers and localized computer networks to communicate with each other. Computer systems connected to a network such as the Internet may be of varying types, e.g., mainframes, workstations, personal computers, etc. The computers are manufactured by different companies using proprietary hardware and operating systems and thus have incompatibilities in their instruction sets, busses, software, file formats and other aspects of their architecture and operating systems. Localized computer networks connected to the Internet may be incompatible with other computer systems and localized networks in terms of the physical layer of communication including the specific hardware used to implement the network. Also, different networks use differing, incompatible protocols for transferring information and are not able to communicate with each other without a translation mechanism such as a "gateway".

The Internet provides a uniform and open standard for allowing various computers and networks to communicate with each other. For example, the Internet uses Transfer Control Protocol/Internet Protocol ("TCP/IP") that defines a uniform packet-switched communication standard which is ultimately used in every transfer of information that takes place over the Internet.

Other Internet standards are the HyperText Transmission Protocol ("HTTP") that allows hypertext documents to be exchanged freely among any computers connected to the Internet and HyperText Markup Language ("HTML") that defines the way in which hypertext documents designate links to information. See, e.g., Berners-Lee, T. J., "The world-wide web," Computer Networks and ISDN Systems 25 (1992).

A hypertext document is a document that allows a user to view a text document displayed on a display device connected to the user's computer and to access, retrieve and view other data objects that are linked to hypertext words or phrases in the hypertext document. In a hypertext document, the user may "click on," or select, certain words or phrases in the text that specify a link to other documents, or data objects. In this way, the user is able to navigate easily among data objects. The data objects may be local to the user's computer system or remotely located over a network. An early hypertext system is Hypercard, by Apple Computer, Inc. Hypercard is a standalone system where the data objects are local to the user's system.

When a user selects a phrase in a hypertext document that has an associated link to another document, the linked document is retrieved and displayed on the user's display screen. This allows the user to obtain more information in an efficient and easy manner. This provides the user with a simple, intuitive and powerful way to "branch off" from a main document to learn more about topics of interest.

Objects may be text, images, sound files, video data, documents or other types of information that is presentable to a user of a computer system. When a document is primarily text and includes links to other data objects according to the hypertext format, the document is said to be a hypertext document. When graphics, sound, video or other media capable of being manipulated and presented in a computer system is used as the object linked to, the document is said to be a hypermedia document. A hypermedia document is similar to a hypertext document, except that the user is able to click on images, sound icons, video icons, etc., that link to other objects of various media types, such as additional graphics, sound, video, text, or hypermedia or hypertext documents.

FIG. **1** shows examples of hypertext and hypermedia documents and links associating data objects in the documents to other data objects. Hypermedia document **10** includes hypertext **20**, an image icon at **22**, a sound icon at **24** and more hypertext **26**. FIG. **1** shows hypermedia document **10** substantially as it would appear on a user's display screen. The user is able to select, or "click" on icons and text on a display screen by using an input device, such as a mouse, in a manner well-known in the art.

When the user clicks on the phrase "hypermedia," software running on the user's computer obtains the link associated with the phrase, symbolically shown by arrow **30**, to access hypermedia document **14**. Hypermedia document **14** is retrieved and displayed on the user's display screen. Thus, the user is presented with more information on the phrase "hypermedia." The mechanism for specifying and locating a linked object such as hypermedia document **14** is an HTML "element" that includes an object address in the format of a Uniform Resource Locator (URL).

Similarly, additional hypertext **26** can be selected by the user to access hypertext document **12** via link **32** as shown in FIG. **1**. If the user selects additional hypertext **26**, then the text for hypertext document **12** is displayed on the user screen. Note that hypertext document **12**, itself, has hypertext at **28**. Thus, the user can click on the phrase "hypermedia" while viewing document **12** to access hypermedia document **14** in a manner similar to that discussed above.

Documents, and other data objects, can be referenced by many links from many different source documents. FIG. **1** shows document **14** serving as a target link for both documents **10** and **12**. A distributed hypertext or hypermedia document typically has many links within it that specify many different data objects located in computers at different geographical locations connected by a network. Hypermedia document **10** includes image icon **22** with a link to image **16**. One method of viewing images is to include an icon, or other indicator, within the text.

Typically, the indicator is a very small image and may be a scaled down version of the full image. The indicator may

be shown embedded within the text when the text is displayed on the display screen. The user may select the indicator to obtain the full image. When the user clicks on image icon **22** browser software executing on the user's computer system retrieves the corresponding full image, e.g., a bit map, and displays it by using external software called a "viewer." This results in the full image, represented by image **16**, being displayed on the screen.

An example of a browser program is the National Center for Supercomputing Application's (NCSA) Mosaic software developed by the University of Illinois at Urbana/Champaign, Ill. Another example is "Cello" available on the Internet at http://www.law.cornell.edu/. Many viewers exist that handle various file formats such as ".TIF," ".GIF," formats. When a browser program invokes a viewer program, the viewer is launched as a separate process. The view displays the full image in a separate "window" (in a windowing environment) or on a separate screen. This means that the browser program is no longer active while the viewer is active. By using indicators to act as place holders for full images that are retrieved and displayed only when a user selects the indicator, data traffic over the network is reduced. Also, since the retrieval and display of large images may require several seconds or more of transfer time the user does not have to wait to have images transferred that are of no interest to the user.

Returning to FIG. **1**, another type of data object is a sound object shown as sound icon **24** within the hypermedia document. When the user selects sound icon **24**, the user's computer accesses sound data shown symbolically by data file **40**. The accessed sound data plays through a speaker or other audio device.

As discussed above, hypermedia documents allow a user to access different data objects. The objects may be text, images, sound files, video, additional documents, etc. As used in this specification, a data object is information capable of being retrieved and presented to a user of a computer system. Some data objects include executable code combined with data. An example of such a combination is a "self-extracting" data object that includes code to "unpack" or decompress data that has been compressed to make it smaller before transferring. When a browser retrieves an object such as a self-extracting data object the browser may allow the user to "launch" the self-extracting data object to automatically execute the unpacking instructions to expand the data object to its original size. Such a combination of executable code and data is limited in that the user can do no more than invoke the code to perform a singular function such as performing the self-extraction after which time the object is a standard data object.

Other existing approaches to embedding interactive program objects in documents include the Object Linking and Embedding (OLE) facility in Microsoft Windows, by Microsoft Corp., and OpenDoc, by Apple Computer, Inc. At least one shortcoming of these approaches is that neither is capable of allowing a user to access embedded interactive program objects in distributed hypermedia documents over networks.

FIG. **2** is an example of a computer network. In FIG. **2**, computer systems are connected to Internet **100**, although in practice Internet **100** may be replaced by any suitable computer network. In FIG. **2**, a user **102** operates a small computer **104**, such as a personal computer or a work station. The user's computer is equipped with various components, such as user input devices (mouse, trackball, keyboard, etc.), a display device (monitor, liquid crystal

display (LCD), etc.), local storage (hard disk drive, etc.), and other components. Typically, small computer **104** is connected to a larger computer, such as server A at **106**. The larger computer may have additional users and computer systems connected to it, such as computer **108** operated by user **110**. Any group of computers may form a localized network. A localized network does not necessarily adopt the uniform protocols of the larger interconnecting network (i.e., Internet **100**) and is more geographically constrained than the larger network. The localized network may connect to the larger network through a "gateway" or "node" implemented on, for example, a server.

Internet **100** connects other localized networks, such as server B at **120**, which interconnects users **122**, **124** and **126** and their respective computer systems to Internet **100**. Internet **100** is made up of many interconnected computer systems and communication links. Communication links may be by hardwire, fiber optic cable, satellite or other radio wave propagation, etc. Data may move from server A to server B through any number of intermediate servers and communication links or other computers and data processing equipment not shown in FIG. **2** but symbolically represented by Internet **100**.

A user at a workstation or personal computer need not connect to the Internet via a larger computer, such as server A or server B. This is shown, for example, by small computer **130** connected directly to Internet **100** as by a telephone modem or other link. Also, a server need not have users connected to it locally, as is shown by server C at **132** of FIG. **2**. Many configurations of large and small computers are possible.

Typically, a computer on the Internet is characterized as either a "client" or "server" depending on the role that the computer is playing with respect to requesting information or providing information. Client computers are computers that typically request information from a server computer which provides the information. For this reason, servers are usually larger and faster machines that have access to many data files, programs, etc., in a large storage associated with the server. However, the role of a server may also be adopted by a smaller machine depending on the transaction. That is, user **110** may request information via their computer **108** from server A. At a later time, server A may make a request for information from computer **108**. In the first case, where computer **108** issues a request for information from server A, computer **108** is a "client" making a request of information from server A. Server A may have the information in a storage device that is local to Server A or server A may have to make requests of other computer systems to obtain the information. User **110** may also request information via their computer **108** from a server, such as server B located at a remote geographical location on the Internet. However, user **110** may also request information from a computer, such as small computer **124**, thus placing small computer **124** in the role of a "server." For purposes of this specification, client and server computers are categorized in terms of their predominant role as either an information requestor or provider. Clients are generally information requestors, while servers are generally information providers.

Referring again to FIG. **1**, data objects such as distributed hypermedia documents **10**, **12** and **14**, image **16** and sound data file **40**, may be located at any of the computers shown in FIG. **2**. Since these data objects may be linked to a document located on another computer the Internet allows for remote object linking.

For example, hypertext document **10** of FIG. **1** may be located at user **110**'s client computer **108**. When user **110**

makes a request by, for example, clicking on hypertext **20** (i.e., the phrase "hypermedia"), user **110**'s small client computer **108** processes links within hypertext document **10** to retrieve document **14**. In this example, we assume that document **14** is stored at a remote location on server B. Thus, in this example, computer **108** issues a command that includes the address of document **14**. This command is routed through server A and Internet **100** and eventually is received by server B. Server B processes the command and locates document **14** on its local storage. Server **14** then transfers a copy of the document back to client **108** via Internet **100** and server A. After client computer **108** receives document **14**, it is displayed so that user **110** may view it.

Similarly, image object **16** and sound data file **40** may reside at any of the computers shown in FIG. **2**. Assuming image object **16** resides on server C when user **110** clicks on image icon **22**, client computer **108** generates a command to retrieve image object **16** to server C. Server C receives the command and transfers a copy of image object **16** to client computer **108**. Alternatively, an object, such as sound data file **40**, may reside on server A so that it is not necessary to traverse long distances via the Internet in order to retrieve the data object.

The Internet is said to provide an "open distributed hypermedia system." It is an "open" system since Internet **100** implements a standard protocol that each of the connecting computer systems, **106**, **130**, **120**, **132** and **134** must implement (TCP/IP). It is a "hypermedia" system because it is able to handle hypermedia documents as described above via standards such as the HTTP and HTML hypertext transmission and mark up standards, respectively. Further, it is a "distributed" system because data objects that are imbedded within a document may be located on many of the computer systems connected to the Internet. An example of an open distributed hypermedia system is the so-called "world-wide web" implemented on the Internet and discussed in papers such as the Berners-Lee reference given above.

The open distributed hypermedia system provided by the Internet allows users to easily access and retrieve different data objects located in remote geographic locations on the Internet. However, this open distributed hypermedia system as it currently exists has shortcomings in that today's large data objects are limited largely by bandwidth constraints in the various communication links in the Internet and localized networks, and by the limited processing power, or computing constraints, of small computer systems normally provided to most users. Large data objects are difficult to update at frame rates fast enough (e.g., 30 frames per second) to achieve smooth animation. Moreover, the processing power needed to perform the calculations to animate such images in real time does not exist on most workstations, not to mention personal computers. Today's browsers and viewers are not capable of performing the computation necessary to generate and render new views of these large data objects in real time.

For example, the Internet's open distributed hypermedia system allows users to view still images. These images are simple non-interactive two-dimensional images, similar to photographs. Much digital data available today exists in the form of high-resolution multi-dimensional image data (e.g., three dimensional images) which is viewed on a computer while allowing the user to perform real time viewing transformations on the data in order for the user to better understand the data.

An example of such type of data is in medical imaging where advanced scanning devices, such as Magnetic Resonance Imaging (MRI) and Computed Tomography (CT), are widely used in the fields of medicine, quality assurance and meteorology to present physicians, technicians and meteorologists with large amounts of data in an efficient way. Because visualization of the data is the best way for a user to grasp the data's meaning, a variety of visualization techniques and real time computer graphics methods have been developed. However, these systems are bandwidth-intensive and compute-intensive and often require multiprocessor arrays and other specialized graphics hardware to carry them out in real time. Also, large amounts of secondary storage for data are required. The expense of these requirements has limited the ability of researchers to readily exchange findings since these larger computers required to store, present and manipulate images are not available to many of the researchers that need to have access to the data.

On the other hand, small client computers in the form of personal computers or workstations such as client computer **108** of FIG. **2** are generally available to a much larger number of researchers. Further, it is common for these smaller computers to be connected to the Internet. Thus, it is desirable to have a system that allows the accessing, display and manipulation of large amounts of data, especially image data, over the Internet to a small, and relatively cheap, client computer.

Due to the relatively low bandwidth of the Internet (as compared to today's large data objects) and the relatively small amount of processing power available at client computers, many valuable tasks performed by computers cannot be performed by users at client computers on the Internet. Also, while the present open distributed hypermedia system on the Internet allows users to locate and retrieve data objects it allows users very little, if any, interaction with these data objects. Users are limited to traditional hypertext and hypermedia forms of selecting linked data objects for retrieval and launching viewers or other forms of external software to have the data objects presented in a comprehensible way.

Thus, it is desirable to have a system that allows a user at a small client computer connected to the Internet to locate, retrieve and manipulate data objects when the data objects are bandwidth-intensive and compute-intensive. Further, it is desirable to allow a user to manipulate data objects in an interactive way to provide the user with a better understanding of information presented and to allow the user to accomplish a wider variety of tasks.

SUMMARY OF THE INVENTION

The present invention provides a method for running embedded program objects in a computer network environment. The method includes the steps of providing at least one client workstation and one network server coupled to the network environment where the network environment is a distributed hypermedia environment; displaying, on the client workstation, a portion of a hypermedia document received over the network from the server, where the hypermedia document includes an embedded controllable application; and interactively controlling the embedded controllable application from the client workstation via communication sent over the distributed hypermedia environment.

The present invention allows a user at a client computer connected to a network to locate, retrieve and manipulate objects in an interactive way. The invention not only allows the user to use a hypermedia format to locate and retrieve program objects, but also allows the user to interact with an

application program located at a remote computer. Interprocess communication between the hypermedia browser and the embedded application program is ongoing after the program object has been launched. The user is able to use a vast amount of computing power beyond that which is contained in the user's client computer.

In one application, high resolution three dimensional images are processed in a distributed manner by several computers located remotely from the user's client computer. This amounts to providing parallel distributed processing for tasks such as volume rendering or three dimensional image transformation and display. Also, the user is able to rotate, scale and otherwise reposition the viewpoint with respect to these images without exiting the hypermedia browser software. The control and interaction of viewing the image may be provided within the same window that the browser is using assuming the environment is a "windowing" environment. The viewing transformation and volume rendering calculations may be performed by remote distributed computer systems.

Once an image representing a new viewpoint is computed the frame image is transmitted over the network to the user's client computer where it is displayed at a designated position within a hypermedia document. By transmitting only enough information to update the image, the need for a high bandwidth data connection is reduced. Compression can be used to further reduce the bandwidth requirements for data transmission.

Progressive

Other applications of the invention are possible. For example, the user can operate a spreadsheet program that is being executed by one or more other computer systems connected via the network to the user's client computer. Once the spreadsheet program has calculated results, the results may be sent over the network to the user's client computer for display to the user. In this way, computer systems located remotely on the network can be used to provide the computing power that may be required for certain tasks and to reduce the data bandwidth by only transmitting results of the computations.

Still other applications of the present invention are possible, as disclosed in the specification, below.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates examples of hypertext and hypermedia documents and links;

FIG. 2 is an example of a computer network;

FIG. 3 is an illustration of a computer system suitable for use with the present invention;

FIG. 4 is an illustration of basic subsystems in the computer system of FIG. 3;

FIG. 5 is an illustration of an embodiment of the invention using a client computer, server computer and a network;

FIG. 6 shows another embodiment of the present invention using additional computers on the network;

FIG. 7A is a flowchart of some of the functionality within the HTMLparse.c file;

FIG. 7B is a flowchart of some of the functionality within the HTMLformat.c file;

FIG. 8A is a flowchart of some of the functionality within the HTMLwidget.c file;

FIG. 8B is a flowchart of some of the functionality within the HTML.c file;

FIG. 9 is a screen display generated in accordance with the present invention; and

FIG. 10 is a diagram of the various processes and data paths in the present invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

375 pages of Source code on 4 microfiche Appendices A and B are provided to this specification. The source code should be consulted to provide details of a specific embodiment of the invention in conjunction with the discussion of the routines in this specification. The source code in Appendix A includes NCSA Mosaic version 2.4 source code along with modifications to the source code to implement the present invention. Appendix B includes source code implementing an application program interface. The source code is written in the "C" computer language to run on an X-Window platform.

FIG. 3 is an illustration of a computer system suitable for use with the present invention. FIG. 3 depicts but one example of many possible computer types or configurations capable of being used with the present invention. FIG. 3 shows computer system **150** including display device **153**, display screen **155**, cabinet **157**, keyboard **159** and mouse **161**. Mouse **161** and keyboard **159** are "user input devices." Other examples of user input devices are a touch screen, light pen, track ball, data glove, etc.

Mouse **161** may have one or more buttons such as buttons **163** shown in FIG. 3. Cabinet **157** houses familiar computer components such as disk drives, a processor, storage means, etc. As used in this specification "storage means" includes any storage device used in connection with a computer system such as disk drives, magnetic tape, solid state memory, bubble memory, etc. Cabinet **157** may include additional hardware such as input/output (I/O) interface cards for connecting computer system **150** to external devices such as an optical character reader, external storage devices, other computers or additional devices.

FIG. 4 is an illustration of basic subsystems in computer system **150** of FIG. 3. In FIG. 4, subsystems are represented by blocks such as central processor **180**, system memory **181** consisting of random access memory (RAM) and/or read-only memory (ROM), display adapter **182**, monitor **183** (equivalent to display device **153** of FIG. 3), etc. The subsystems are interconnected via a system bus **184**. Additional subsystems such as a printer, keyboard, fixed disk and others are shown. Peripherals and input/output (I/O) devices can be connected to the computer system by, for example serial port **185**. For example, serial port **185** can be used to connect the computer system to a modem for connection to a network or serial port **185** can be used to interface with a mouse input device. The interconnection via system bus **184** allows central processor **180** to communicate with each subsystem and to control the execution of instructions from system memory **181** or fixed disk **186**, and the exchange of information between subsystems. Other arrangements of subsystems and interconnections are possible.

FIG. 5 is an illustration of an embodiment of the invention using a client computer, server computer and a network.

In FIG. 5, client computer **200** communicates with server computer **204** via network **206**. Both client computer **200** and server computer **204** use a network protocol layer to communicate with network **206**. In a preferred embodiment, network **206** is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used. For ease of illustration, additional hardware and software layers are not shown in FIG. 5.

Client computer **200** includes processes, such as browser client **208** and application client **210**. In a preferred

embodiment, application client **210** is resident within client computer **200** prior to browser client **208**'s parsing of a hypermedia document as discussed below. In a preferred embodiment application client **210** resides on the hard disk or RAM of client computer **200** and is loaded (if necessary) and executed when browser client **208** detects a link to application client **210**. The preferred embodiment uses the XEvent interprocess communication protocol to exchange information between browser client **208** and application client **210** as described in more detail, below. Another possibility is to install application client **210** as a "terminate and stay resident" (TSR) program in an operating system environment, such as X-Window. Thereby making access to application client **210** much faster.

Browser client **208** is a process that a user of client computer **200** invokes in order to access various data objects, such as hypermedia documents, on network **206**. Hypermedia document **212** shown within client computer **200** is an example of a hypermedia document, or object, that a user has requested access to. In this example, hypermedia document **212** has been retrieved from a server connected to network **206** and has been loaded into, e.g., client computer **200**'s RAM or other storage device.

Once hypermedia document **212** has been loaded into client computer **200**, browser client **208** parses hypermedia document **212**. In parsing hypermedia document **212**, browser client **208** detects links to data objects as discussed above in the Background of the Invention section. In FIG. **5**, hypermedia document **212** includes an embedded program link at **214**. Embedded program link **214** identifies application client **212** as an application to invoke. In this present example, the application, namely, application client **210**, resides on the same computer as the browser client **208** that the user is executing to view the hypermedia document. Embedded program link **214** may include additional information, such as parameters, that tell application client **210** how to proceed. For example, embedded program link **214** may include a specification as to a data object that application client **210** is to retrieve and process.

When browser client **208** encounters embedded program link **214**, it invokes application client **210** (optionally, with parameters or other information) and application client **210** executes instructions to perform processing in accordance with the present invention.

An example of the type of processing that application client **210** may perform is multidimensional image visualization. Note that application client **210** is in communication with network **206** via the network protocol layer of client computer **200**. This means that application client **210** can make requests over network **206** for data objects, such as multidimensional image objects. For example, application client **210** may request an object, such as object 1 at **216**, located in server computer **204**. Application client **210** may make the request by any suitable means. Assuming network **206** is the Internet, such a request would typically be made by using HTTP in response to a HTML-style link definition for embedded program link **214**.

Assuming application client **210** has made a request for the data object at **216**, server process **218** ultimately receives the request. Server process **218** then retrieves data object **216** and transfers it over network **206** back to application client **210**. To continue with the example of a multidimensional visualization application, data object **216** may be a three dimensional view of medical data for, e.g., an embryo.

After application client **210** receives the multidimensional data object **216**, application client **210** executes instructions to display the multidimensional embryo data on the display screen to a user of the client computer **200**. The user is then able to interactively operate controls to recompute different views for the image data. In a preferred embodiment, a control window is displayed within, or adjacent to, a window generated by browser client **208** that contains a display of hypermedia document **212**. An example of such display is discussed below in connection with FIG. **9**. Thus, the user is able to interactively manipulate a multidimensional image object by means of the present invention. In order to make application client **210** integral with displays created by browser client **208**, both the browser client and the application client must be in communication with each other, as shown by the arrow connecting the two within client computer **200**. The manner of communication is through an application program interface (API), discussed below.

Browser client **208** is a process, such as NCSA Mosaic, Cello, etc. Application client **210** is embodied in software presently under development called "VIS" and "Panel" created by the Center for Knowledge Management at the University of California, San Francisco, as part of the Doyle Group's distributed hypermedia object embedding approach described in "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web," by C. Ang, D. Martin, M. Doyle; to be published in the Proceedings of Visualization 1994, IEEE Press, Washington, D.C., October 1994.

Versions and descriptions of software embodying the present invention are generally available as hyperlinked data objects from the Visible Embryo Project's World Wide Web document at the URL address "HTTP://visembryo.ucsf.edu/".

Another embodiment of the present invention uses an application server process executing on server computer **204** to assist in processing that may need to be performed by an external program. For example, in FIG. **5**, application server **220** resides on server computer **204**. Application server **220** works in communication with application client **210** residing on client computer **200**. In a preferred embodiment, application server **220** is called VRServer, also a part of Doyle Group's approach. Since server computer **204** is typically a larger computer having more data processing capabilities and larger storage capacity, application server **220** can operate more efficiently, and much faster, than application client **210** in executing complicated and numerous instructions.

In the present example where a multidimensional image object representing medical data for an embryo is being viewed, application server **220** could perform much of the viewing transformation and volume rendering calculations to allow a user to interactively view the embryo data at their client computer display screen. In a preferred embodiment, application client **210** receives signals from a user input device at the user's client computer **200**. An example of such input would be to rotate the embryo image from a current position to a new position from the user's point of view. This information is received by application client **210** and processed to generate a command sent over network **206** to application server **220**. Once application server **220** receives the information in the form of, e.g., a coordinate transformation for a new viewing position, application server **220** performs the mathematical calculations to compute a new view for the embryo image. Once the new view has been computed, the image data for the new view is sent over network **206** to application client **210** so that application client **210** can update the viewing window currently displaying the embryo image. In a preferred embodiment,

prop

application server 220 computes a frame buffer of raster display data, e.g., pixel values, and transfers this frame buffer to application client 210. Techniques, such as data compression and delta encoding, can be used to compress the data before transmitting over network 206 to reduce the bandwidth requirement.

It will be readily seen that application server 220 can advantageously use server computer 204's computing resources to perform the viewing transformation much more quickly than could application client 210 executing on client computer 200. Further, by only transmitting the updated frame buffer containing a new view for the embryo image, the amount of data sent over network 206 is reduced. By using appropriate compression techniques, such as, e.g., MPEG (Motion Picture Experts Group) or JPEG (Joint Photographic Experts Group), efficient use of network 206 is preserved.

FIG. 6 shows yet another embodiment of the present invention. FIG. 6 is similar to FIG. 5, except that additional computers 222 and 224 are illustrated. Each additional computer includes a process labeled "Application (Distributed)." The distributed application performs a portion of the task that an application, such as application server 220 or application client 210, perform. In the present. example, tasks such as volume rendering may be broken up and easily performed among two or more computers. These computers can be remote from each other on network 206. Thus, several computers, such as server computer 204 and additional computers 222 and 224 can all work together to perform the task of computing a new viewpoint and frame buffer for the embryo for the new orientation of the embryo image in the present example. The coordination of the distributed processing can be performed at client computer 200 by application client 210, at server computer 204 by application server 220, or by any of the distributed applications executing on additional computers, such as 222 and 224. In a preferred embodiment, distributed processing is coordinated by a program called "VIS" represented by application client 210 in FIG. 6.

Other applications of the invention are possible. For example, the user can operate a spreadsheet program that is being executed by one or more other computer systems connected via the network to the user's client computer. Once the spreadsheet program has calculated results, those results may be sent over the network to the user's client computer for display within the hypermedia document on the user's client computer. In this way, computer systems located remotely on the network can be used to provide the computing power that may be required for certain tasks and to reduce the data bandwidth required by only transmitting results of the computations.

Another type of possible application of this invention would involve embedding a program which runs only on the client machine, but which provides the user with more functionality than exists in the hypermedia browser alone. An example of this is an embedded client application which is capable of viewing and interacting with images which have been processed with Dr. Doyle's MetaMAP invention (U.S. Pat. No. 4,847,604). This MetaMAP process uses object-oriented color map processing to allow individual color index ranges within paletted images to have object identities, and is useful for the creation of, for example, interactive picture atlases. It is a more efficient means for defining irregular "hotspots" on images than the ISMAP function of the World Wide Web, which uses polygonal outlines to define objects in images. A MetaMAP-capable client-based image browser application can be embedded, together with an associated image, within a hypermedia document, allowing objects within the MetaMAP-processed image to have URL addresses associated with them. When a user clicks with a mouse upon an object within the MetaMAP-processed image, the MetaMAP client application relays the relevant URL back to the hypermedia browser application, which then retrieves the HTML file or hypermedia object which corresponds to that URL.

The various processes in the system of the present invention communicate through a custom API called Mosaic/External Application Program Interface MEAPI. The MEAPI set of predefined messages includes those shown in Table I.

TABLE I

Message Function	Message Name
Messages from server to client:	
1. Server Update Done	XtNrefreshNotify
2. Server Ready	XtNpanelStartNotify
3. Server Exiting	XtNpanelExitNotify
Messages from client to server:	
4. Area Shown	XtNmapNotify
5. Area Hidden	XtNunmapNotify
6. Area Destroyed	XtNexitNotify

The messages in Table I are defined in the file protocol_lib.h in Appendix B. The functions of the MEAPI are provided in protocol_lib.c of Appendix B. Thus, by using MEAPI a server process communicates to a client application program to let the client application know when the server has finished updating information, such as an image frame buffer, or pixmap (Message 1); when the server is ready to start processing messages (Message 2) and when the server is exiting or stopping computation related to the server application program.

For client to server communications, MEAPI provides for the client informing the server when the image display window area is visible, when the area is hidden and when the area is destroyed. Such information allows the server to decide whether to allocate computing resources for, e.g., rendering and viewing transformation tasks where the server is running an application program to generate new views of a multi dimensional object. Source code for MEAPI fundamental functions such as handle_client_msg, register_client, register_client_msg_callback and send_client_msg may be found in protocol_lib.c as part of the source code in Appendix B.

Next, a discussion of the software processes that perform parsing of a hypermedia document and launching of an application program is provided in connection with Table II and FIGS. 7A, 7B, 8A and 8B.

Table II, below, shows an example of an HTML tag format used by the present invention to embed a link to an application program within a hypermedia document.

TABLE II

```
<EMBED
    TYPE = "type"
    HREF = "href"
    WIDTH = width
    HEIGHT = height
>
```

As shown in Table II, the EMBED tag includes TYPE, HREF, WIDTH and HEIGHT elements. The TYPE element

is a Multipurpose Internet Mail Extensions (MIME) type. Examples of values for the TYPE element are "application/x-vis" or "video/mpeg". The type "application /x-vis" indicates that an application named "x-vis" is to be used to handle the object at the URL specified by the HREF. Other types are possible such as "application/x-inventor", "application/postscript" etc. In the case where TYPE is "application/x-vis" this means that the object at the URL address is a three dimensional image object since the program "x-vis" is a data visualization tool designed to operate on three dimensional image objects. However, any manner of application program may be specified by the TYPE element so that other types of applications, such as a spreadsheet program, database program, word processor, etc. may be used with the present invention. Accordingly, the object reference by the HREF element would be, respectively, ;a spreadsheet object, database object, word processor document object, etc.

On the other hand, TYPE values such as "video/mpeg", "image/gif", "video/x-sgi-movie", etc. describe the type of data that HREF specifies. This is useful where an external application program, such as a video player, needs to know what format the data is in, or where the browser client needs to determine which application to launch based on the data format. Thus, the TYPE value can specify either an application program or a data type. Other TYPE values are possible. HREF specifies a URL address as discussed above for a data object. Where TYPE is "application/x-vis" the URL address specifies a multi-dimensional image object. Where TYPE is "video/mpeg" the URL address specifies a video object.

WIDTH and HEIGHT elements specify the width and height dimensions, respectively, of a Distributed Hypermedia Object Embedding (DHOE) window to display an external application object such as the three dimensional image object or video object discussed above.

FIG. 7A is a flowchart describing some of the functionality within the HTMLparse.c file of routines. The routines in HTMLparse.c perform the task of parsing a hypermedia document and detecting the EMBED tag. In a preferred embodiment, the enhancements to include the EMBED tag are made to an HTML library included in public domain NCSA Mosaic version 2.4. Note that much of the source code in is pre-existing NCSA Mosaic code. Only those portions of the source code that relate to the new functionality discussed in this specification should be considered as part of the invention. The new functionality is identifiable as being set off from the main body of source code by conditional compilation macros such as "#ifdef . . . #endif" as will be readily apparent to one of skill in the art.

In general, the flowcharts in this specification illustrate one or more software routines executing in a computer system such as computer system **1** of FIG. **1**. The routines may be implemented by any means as is known in the art. For example, any number of computer programming languages, such as "C", Pascal, FORTRAN, assembly language, etc., may be used. Further, various programming approaches such as procedural, object oriented or artificial intelligence techniques may be employed.

The steps of the flowcharts may be implemented by one or more software routines, processes, subroutines, modules, etc. It will be apparent that each flowchart is illustrative of merely the broad logical flow of the method of the present invention and that steps may be added to, or taken away from, the flowcharts without departing from the scope of the invention. Further, the order of execution of steps in the flowcharts may be changed without departing from the scope of the invention. Additional considerations in implementing the method described by the flowchart in software may dictate changes in the selection and order of steps. Some considerations are event handling by interrupt driven, polled, or other schemes. A multiprocessing or multitasking environment could allow steps to be executed "concurrently." For ease of discussion the implementation of each flowchart may be referred to as if implemented in a single "routine".

The modifications to NCSA Mosaic version 2.4 software files HTMLparse.c, HTMLformat.c, HTMLwidget.c and HTML.c will next be discussed, in turn.

Returning to FIG. **7**, it is assumed that a hypermedia document has been obtained at a user's client computer and that a browser program executing on the client computer displays the document and calls a first routine in the HTMLparse.c file called "HTMLparse". This first routine, HTMLparse, is entered at step **252** where a pointer to the start of the document portion is passed. Steps **254**, **256** and **258** represent a loop where the document is parsed or scanned for HTML tags or other symbols. While the file HTMLparse.c includes routines to handle all possible tags and symbols that may be encountered, FIG. **7A**, for simplicity, only illustrates the handling of EMBED tags.

Assuming there is more text to parse, execution proceeds to step **256** where routines in HTMLparse.c obtain the next item (e.g., word, tag or symbol) from the document. At step **258** a check is made as to whether the current tag is the EMBED tag. If not, execution returns to step **254** where the next tag in the document is obtained. If, at step **258**, it is determined that the tag is the EMBED tag, execution proceeds to step **260** where an enumerated type is assigned for the tag. Each occurrence of a valid EMBED tag specifies an embedded object. HTMLParse calls a routine "get_mark" in HTMLparse.c to put sections of HTML document text into a "markup" text data structure. Routine get_mark, in turn, calls ParseMarkType to assign an enumerated type. The enumerated type is an identifier with a unique integer associated with it that is used in later processing described below.

Once all of the hypermedia text in the text portion to be displayed has been parsed, execution of HTMLparse.c routines terminates at step **262**.

FIG. **7B** is a flowchart of routines in file HTMLformat.c to process the enumerated type created for the EMBED tag by routines in HTMLparse.c. In the X-Window implementation of a preferred embodiment, the enumerated type is processed as if it is a regular Motif/XT widget. For details on X-Window development see, e.g., "Xlib Programming Manual," "X Toolkit Intrinsics Programming Manual" and "Motif Programming Manual" published by O'Reilly & Associates, Inc. HTMLformat is entered at step **270** where a pointer to the enumerated type to process is passed.

At step **272** the parameters of the structure are initialized in preparation for inserting a DrawingArea widget on an HTML page. This includes providing values for the width and height of a window on the display to contain an image, position of the window, style, URL of the image object, etc. Various codes are also added by routines in HTMLformat.c (such as TriggerMarkChanges) to insert an internal representation of the HTML statement into an object list maintained internally by the browser. In the X-Window application corresponding to the source code of Appendix A, the browser is NCSA Mosaic version 2.4.

FIG. **8A** is a flowchart for routine HTMLwidget. HTMLwidget creates display data structures and launches an external application program to handle the data object specified by the URL in the EMBED tag.

HTMLwidget is entered at step **280** after HTMLformat has created the internal object representation of the EMBED tag. HTMLwidget is passed the internal object and performs its processing on the object. At step **282** the DrawingArea widget is created according to the type of the internal representation, from HTMLformat, specified in the internal object. Similarly, at step **284** a pixmap area for backing storage is defined.

At step **286** a check is made as to whether the type attribute of the object, i.e., the value for the TYPE element of the EMBED tag, is an application. If so, step **290** is executed to launch a predetermined application. In a preferred embodiment an application is launched according to a user-defined list of application type/application pairs. The list is defined as a user-configurable XResource as described in "Xlib Programming Manual." An alternative embodiment could use the MIME database as the source of the list of application type/application pairs. The routine "vis_start_external_application" in file HTMLformat.c is invoked to match the application type and to identify the application to launch.

The external application is started as a child process of the current running process (Mosaic), and informed about the window ID of the DrawingArea created in HTMLformat. The external application is also passed information about the ID of the pixmap, the data URL and dimensions. Codes for communication such as popping-up/iconifying, start notification, quit notification and refresh notification with external applications and DrawingArea refreshing are also added. Examples of such codes are (1) "setup/start" in vis_register_client and vis_get panel_window in HTMLwidgets.c; (2) "handle messages from external applications" in vis_handle panel_msg in HTMLwidgets.c; (3) "send messages to external applications" in vis_send_msg in HTMLwidgets.c; (4) "terminate external applications" in vis_exit in HTMLwidgets.c which calls vis_send_msg to send a quit message; and (5) "respond to refresh msgs" in vis_redraw in HTMLwidgets.c.

If, at step **286**, the type is determined not to be an application object (e.g., a three dimensional image object in the case of application "x-vis") a check is made at step **288** to determine if the type is a video object. If so, step **292** is executed to launch a video player application. Parameters are passed to the video player application to allow the player to display the video object within the DrawingArea within the display of the portion of hypermedia document on the client's computer. Note that many other application objects types are possible as described above.

FIG. **8**B is a flowchart for routine HTML. Routine HTML takes care of "shutting down" the objects, data areas, etc. that were set up to launch the external application and display the data object. HTML is entered at step **300** and is called when the display or other processing of the EMBED tag has been completed. At step **302** the display window is removed and the memory areas for the pixmap and internal object structure is made free for other uses. Completion of processing can be by user command or by computer control.

The present invention allows a user to have interactive control over application objects such as three dimensional image objects and video objects. In a preferred embodiment, controls are provided on the external applications' user interface. In the case of a VIS/panel application, a process, "panel" creates a graphical user interface (GUI) thru which the user interacts with the data. The application program, VIS, can be executing locally with the user's computer or remotely on a server, or on one or more different computers, on the network. The application program updates pixmap data and transfers the pixmap data (frame image data) to a buffer to which the browser has access. The browser only needs to respond to the refresh request to copy the contents from the updated pixmap to the DrawingArea. The Panel process sends messages as "Msg" sending performed by routines such as vis_send_msg and vis_handle panel_msg to send events (mousemove, keypress, etc.) to the external application.

FIG. **9** is a screen display of the invention showing an interactive application object (in this case a three dimensional image object) in a window within a browser window. In FIG. **9**, the browser is NCSA Mosaic version 2.4. The processes VIS, Panel and VRServer work as discussed above. FIG. **9** shows screen display **356** Mosaic window **350** containing image window **352** and a portion of a panel window **354**. Note that image window **352** is within Mosaic window **350** while panel window **354** is external to Mosaic window **350**. Another possibility is to have panel window **354** within Mosaic window **350**. By using the controls in panel window **354** the user is able to manipulate the image within image window **352** in real time do perform such operations as scaling, rotation, translation, color map selection, etc. In FIG. **9**, two Mosaic windows are being used to show two different views of an embryo image. One of the views is rotated by six degrees from the other view so that a stereoscopic effect can be achieved when viewing the images. Communication between Panel and VIS is via "Tooltalk" described in, e.g., "Tooltalk 1.1.1 Reference Manual," from SunSoft.

FIG. **10** is an illustration of the processes VIS, Panel and VRServer discussed above. As shown in FIG. **10**, the browser process, Mosaic, communicates with the Panel process via inter-client communication mechanisms such as provided in the X-Window environment. The Panel process communicates with the VIS process through a communications protocol (ToolTalk, in the preferred embodiment) to exchange visualization command messages and image data. The image data is computed by one or more copies of a process called VRServer that may be executing on remote computers on the network. VRServer processes respond to requests such as rendering requests to generate image segments. The image segments are sent to VIS and combined into a pixmap, or frame image, by VIS. The frame image is then transferred to the Mosaic screen via communications between VIS, Panel and Mosaic. A further description of the data transfer may be found in the paper "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web," referenced above.

In the foregoing specification, the invention has been described with reference to a specific exemplary embodiment thereof. It will, however, be evident that various modifications and changes may be made thereunto without departing from the broader spirit and scope of the invention as set forth in the appended claims. For example, various programming languages and techniques can be used to implement the disclosed invention. Also, the specific logic presented to accomplish tasks within the present invention may be modified without departing from the scope of the invention. Many such changes or modifications will be readily apparent to one of ordinary skill in the art. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense, the invention being limited only by the provided claims.

What is claimed is:

1. A method for running an application program in a computer network environment, comprising:

providing at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;

executing, at said client workstation, a browser application, that parses a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and for responding to predetermined text formats to initiate processing specified by said text formats; utilizing said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document, wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document, and wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable interactive processing of said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.

2. The method of claim 1, wherein said executable application is a controllable application and further comprising the step of:

interactively controlling said controllable application on said client workstation via inter-process communications between said browser and said controllable application.

3. The method of claim 2, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.

4. The method of claim 3, wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes the following substeps:

issuing, from the client workstation, one or more commands to the network server;

executing, on the network server, one or more instructions in response to said commands;

sending information from said network server to said client workstation in response to said executed instructions; and processing said information at the client workstation to interactively control said controllable application.

5. The method of claim 4, wherein said additional instructions for controlling said controllable application reside on said client workstation.

6. A computer program product for use in a system having at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment, the computer program product comprising:

a computer usable medium having computer readable program code physically embodied therein, said computer program product further comprising:

computer readable program code for causing said client workstation to execute a browser application to parse a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and to respond to predetermined text formats to initiate processes specified by said text formats;

computer readable program code for causing said client workstation to utilize said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document, wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document, and wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable interactive processing of said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.

7. The computer program product of claim 6, wherein said executable application is a controllable application and further comprising:

computer readable program code for causing said client workstation to interactively control said controllable application on said client workstation via inter-process communications between said browser and said controllable application.

8. The computer program product of claim 7, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.

9. The computer program product of claim 8, wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes:

computer readable program code for causing said client workstation to issue, from the client workstation, one or more commands to the network server;

computer readable program code for causing said network server to execute one or more instructions in response to said commands;

computer readable program code for causing said network sever to send information to said client workstation in response to said executed instructions; and

computer readable program code for causing said client workstation to process said information at the client workstation to interactively control said controllable application.

10. The computer program product of claim 9, wherein said additional instructions for controlling said controllable application reside on said client workstation.

* * * * *